In [71]:
```
# first we are going to import neccessary library
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
    endpoint_517c4add7ff840f8915f46a7a0f89295 = 'https://s3.ap.cloud-object-storage.appdomain.cloud'
else:
    endpoint_517c4add7ff840f8915f46a7a0f89295 = 'https://s3.private.ap.cloud-object-storage.appdomain.cloud'

client_517c4add7ff840f8915f46a7a0f89295 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jrUiIrk68Cc8AeuNY0kkYbwC-HP2b_RQ2y905E3I6Wvu',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_517c4add7ff840f8915f46a7a0f89295)

body = client_517c4add7ff840f8915f46a7a0f89295.get_object(Bucket='pythonprojectfordatascience-donotdelete-pr-i4fyreja3rirtd',Key='Sacramentorealestatetra
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_1 = pd.read_csv(body)
df_data_1.head()

# call the data
```

Out[2]:

	street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.519470	-121.435768

```
# finding index number from each city
pd.DataFrame(df_data_1['city'].unique())
```

	0
0	SACRAMENTO
1	RANCHO CORDOVA
2	RIO LINDA
3	CITRUS HEIGHTS
4	NORTH HIGHLANDS
5	ANTELOPE
6	ELK GROVE
7	ELVERTA
8	GALT
9	CARMICHAEL
10	ORANGEVALE
11	FOLSOM
12	MATHER
13	POLLOCK PINES
14	GOLD RIVER
15	EL DORADO HILLS
16	RANCHO MURIETA
17	WILTON
18	GREENWOOD
19	FAIR OAKS
20	CAMERON PARK
21	LINCOLN
22	PLACERVILLE

23	MEADOW VISTA
24	ROSEVILLE
25	ROCKLIN
26	AUBURN
27	LOOMIS
28	EL DORADO
29	PENRYN
30	GRANITE BAY
31	FORESTHILL
32	DIAMOND SPRINGS
33	SHINGLE SPRINGS
34	COOL
35	WALNUT GROVE
36	GARDEN VALLEY
37	SLOUGHHOUSE
38	WEST SACRAMENTO

In [14]:
```
# finding unique data from baths
set(df_data_1['baths'])
```

Out[14]: {0, 1, 2, 3, 4, 5}

In [15]:
```
# call the ML library
labelencoder = LabelEncoder()
```

In [16]:
```
# change all the dtring data from city into number
df_data_1['city'] = labelencoder.fit_transform(df_data_1['city'])
```

In [17]:
```
# finding all index number from each city
pd.DataFrame(set(df_data_1['type']))
```

Out[17]:

	0
0	Condo
1	Unkown
2	Multi-Family
3	Residential

In [18]:
```
# changing string data from type column into number data
df_data_1['type'] = labelencoder.fit_transform(df_data_1['type'])
```

In [19]:
```
# making train dataset and test dataset
x1 = df_data_1[['city','beds','baths','sq__ft','price']]
```

In [20]:
```
y1 = df_data_1['type']
```

In [21]:
```
x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1, test_size = 0.1, random_state = 10)
```

In [22]:
```
# call the ML library
dtc = DecisionTreeClassifier()
```

In [23]:
```
dtc.fit(x1_train, y1_train)
```

Out[23]:
```
DecisionTreeClassifier()
```

In [24]:
```
x1predict = dtc.predict(x1_test)
x1predict
```

Out[24]:
```
array([2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [25]:
```
accuracy_score(dtc.predict(x1_test), y1_test)
```

Out[25]:
```
0.9292929292929293
```

In [27]:
```
cf_matrix =  confusion_matrix(dtc.predict(x1_test), y1_test)
```

In [74]:
```
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot = True, cmap = 'Reds')
plt.xlabel('test value')
plt.ylabel('actual value')
```

Out[74]: Text(33.0, 0.5, 'actual value')